A Multi-threaded Version of Field II

Jensen, Jørgen Arendt

Published in:
Proceedings of IEEE International Ultrasonics Symposium

Link to article, DOI:
10.1109/ULTSYM.2014.0555

Publication date:
2014

Document Version
Early version, also known as pre-print

Paper presented at the IEEE International Ultrasonics Symposium, Chicago, Il., USA, 2014:

A Multi-threaded Version of Field II

Jørgen Arendt Jensen

Center for Fast Ultrasound Imaging,
Biomedical Engineering group, Department of Electrical Engineering, Bldg. 349,
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

To be published in Proceedings of IEEE International Ultrasonics Symposium, Chicago, Il., USA, 2014.

A Multi-threaded Version of Field II

Jørgen Arendt Jensen

Center for Fast Ultrasound Imaging, Department of Electrical Engineering,
Technical University of Denmark, DK-2800 Lyngby, Denmark

***Abstract*—A multi-threaded version of Field II has been developed, which automatically can use the multi-core capabilities of modern CPUs. The memory allocation routines were rewritten to minimize the number of dynamic allocations and to make pre-allocations possible for each thread. This ensures that the simulation job can be automatically partitioned and the interdependence between threads minimized. The new code has been compared to Field II version 3.22, October 27, 2013 (latest free-ware version). A 64 element 5 MHz focused array transducer was simulated. One million point scatterers randomly distributed in a plane of 20 x 50 mm (width x depth) with random Gaussian amplitudes were simulated using the command *calc_scat*. Dual Intel Xeon CPU E5-2630 2.60 GHz CPUs were used under Ubuntu Linux 10.02 and Matlab version 2013b. Each CPU holds 6 cores with hyper-threading, corresponding to a total of 24 hyper-threading cores. The averaged simulation time for 10 realizations for the old version was 85.1 s. A single thread run for the new version took 27.7 s; a speed-up of 3.1. Employing all 24 cores gave a simulation time of 3.27 s for the one million scatterers corresponding to a speed-up factor of 26 times. The speed-up in general depends on the transducer, scatterers and simulation, and it varies across applications between 13 and 30. The program is fully compatible with older versions, and only a single command has been added for setting the number of threads to use. The division of labor is automatically handled by the program. For a phantom with 100,000 scatterers, it is now possible to simulate a full 128 line image in around 42 seconds with full precision.**

I. Introduction

Simulation of new imaging schemes and estimation methods is now the first step in their development. This often entails Monte Carlo simulations for a range of parameter values involving a large collection of scatterers. This inevitably leads to long simulation times for the many channels in modern array transducers. Many groups have devised simulation solutions. This includes Ultrasim from Sverre Holm's group [1] and the program in D'hooge's group [2] both of which are based on spatial impulse responses. An accelerated version based one GPU implementation has also been made [3]. Another GPU solution was developed in [4], which is based on convolving a point spread function with a scatterer map. This in general gives a very fast solution, but demands specialized hardware and does not fully model the spatial variation and complexity of ultrasound fields.

Another solution is to use the angular spectrum approach as in the Focus program [5]. This use the continuous wave solution to calculate the field in a plane. This is a fast approach, but many frequency components have to be summed to get the temporal response, and the fixed plane can be restrictive for flow simulations.

A more general solution using finite element modeling was developed by Pinton and Trahey [6]. This can also include inhomogeneous media and non-linear propagation, but leads to very long simulation times (days on computer clusters) making it less ideal for parameter optimization. A general k-space approach was developed in [7], but still has the problem of very long simulation times.

The most widely used simulation code is Field II [8], [9], which has been cited more than 690 times (ISI Web of science, August, 2014). It has been used in numerous investigations of advanced imaging techniques. It is based on spatial impulse responses [10], [11], [12]. This is a general method for any linear simulation and has shown to be a very accurate solution in numerous studies. The initial version of the Field program was developed in 1990 on an Apollo DN3000 workstation with a peak calculation speed of 72 kFlOPS and very limited RAM resources. Modern processors have access to Gbytes of RAM and can yield around 10-100 GFLOPS. This made it possible to use Field II in Monte Carlo simulations of very large parameter studies for e.g. 3-D imaging, vector velocity imaging, and other applications. The studies, however, often take a significant amount of CPU time, and have to be conducted using many invocations of Matlab in parallel on cluster computers. This often also entails writing programs with synchronization between the different invocations. The purpose of this work is to develop a multi-threaded version of Field II, which uses all the benefits of modern CPUs to reduce computation time.

Older workstations only contained one CPU, and RAM access was fast compared to the time for executing floating point operations. Current CPUs hold multiple computing cores, and the RAM access time is large compared to the time for one floating point calculation. They also house generous caches for speeding up data access. This is used in a new multi-threaded version of Field II that uses dynamic partition of the calculation. The memory allocation routines were rewritten to minimize the number of dynamic allocations to increase cache usage and to make pre-allocations possible for each thread. This ensures that the simulation job can be automatically partitioned and the interdependence between threads minimized. The principles behind the design is discussed in Section II. Results from using the code are given in Section III and discussed in Section IV.

II. Computer architectures

Simulation of ultrasound imaging is inherently parallel. The arrays consists of many elements, scattering emanates from a large and independent collection of scatterers, imaging is often done in many independent directions, and often simulations also have to be conducted over time for *e.g.* flow imaging. There are, thus, many ways in which simulation jobs can be partitioned.

Traditionally the division has been over imaging lines and time [13], [14]. One job is made for one imaging direction and the resulting signal stored in a file. Several computers can then automatically partition the work by using a shared disk storage and store a dummy file before the job is executed. The existence of a result file indicates that the line has been simulated or that another computer is taking care of the job. This works for jobs running from minutes to hours with limited race conditions on storing the file. This could also be handled automatically through batch systems and pre-allocation of resources, although it demands some overhead and a mechanism for handling non-completed jobs.

Another approach is to simulate all combinations of transmit and receive signals as in [14]. The beamforming is then not included in the simulation and is performed afterwards. The approach has shown to give roughly a factor 6 times faster simulation time, but at the cost of having to perform the beamforming afterwards. It also uses quite some memory as N^2 signals have to be made for a transducer with N elements. The speed-up is generated from only simulating the signal from each scatterer one time and not for every imaging line.

The major drawback of these methods is that the overall simulation time are still significant before the first results are found, and that parallel programs have to be made for the simulation. The purpose of this paper is to demonstrate how a parallel version of Field II can speed up the execution time without the user making modifications to his code. The implementation also more efficiently makes use of the memory and cache in the computer as only one instance of Matlab is used.

The architecture of a modern multi-core CPU is shown in Fig. 1. The processor consists of 8 cores with a small L1+L2 cache that can house a limited amount of data. The caches are used for having a very fast access to the program and data. A shared L3 cache is also associated with the processors. It is, thus, important that the data being processed can be housed inside the caches as cache misses often have a very high overhead from the fetching of data from RAM.

Field II principally uses memory for two objects: The first object is the parametric description of the transducer and its function in terms of impulse response, apodizations and focusing. The second object is dynamically allocated signals. These are used for the calculated signals like spatial impulse responses and received signals. A new signal is allocated every time a spatial impulse response from a transducer element is found or a number of signals combined. A typical calculation is performed for 192 element transducer arrays for

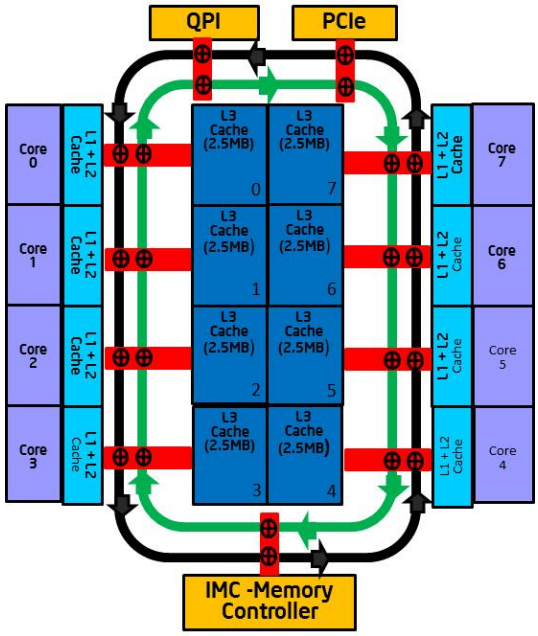

Fig. 1. Cache architecture of Intel Xeon processor with 8 cores (taken from https://software.intel.com/en-us/articles/intel-xeon-processor-e5-26004600-product-family-technical-overview).

1 million point scatterers. This will entail the allocation of $2 \times 1 \cdot 10^6 \times 192 = 384 \cdot 10^6$ signals for each image line. It is a key aspect of Field II that this is dynamic to avoid having restrictions on transducer size, number of scatterers, or length of signals. This has allowed the program to also simulate large 2-D arrays and large phantoms. Allocating new memory, however, often has a high overhead in computer systems and should in general be minimized.

The approach taken here is to have a common description of the transducer, which is used by all threads. It will therefore quickly be placed in the L3 cache. The dynamic signal allocation is maintained, but the memory of unused signals are not released to the operating system, but are put on a queue. Requesting a new signal will therefore most likely give a signal already residing in the cache memory. Also pre-allocation of internal variables, that can be used, has been optimized in the code. Therefore allocation is often done outside the most computationally intense routines to minimize allocations. These three approaches have made it possible to make an implementation that nearly scales with the number of cores.

III. Simulation Method and Results

The new code has been run on a Dual Intel Xeon® E5-2630 2.60 GHz CPU with 32 Gbytes RAM under Ubuntu Linux and Matlab version 2013b. Each processor has 6 real cores that are hyper-threaded to behave like 12 cores for a single CPU in total giving 24 hyper-threaded cores. Hyper-threaded cores often give an increase in performance as the utilization of the core is increased, but it should be emphasized that hyper-threading does not give two independent CPU cores in one physical core.

TABLE I
SIMULATION PARAMETERS USED FOR THE PERFORMANCE ANALYSIS.

Sampling frequency	f_s	100	MHz
Speed of sound	c	1540	m/s
Transducer center frequency	f_0	5	MHz
Number of cycles in pulse	M	2	
Wavelength lambda	$\lambda = c/f_0$	0.308	mm
Width of element	$\lambda/2$	0.154	mm
Height of element	h	5	mm
Kerf	$k_e = \lambda/10$	0.031	mm
Fixed focal point	f_c	[0 0 60]	mm
Number of physical elements	N_e	64	
Pulse duration	$T_p = M/f_0$	0.4	μs

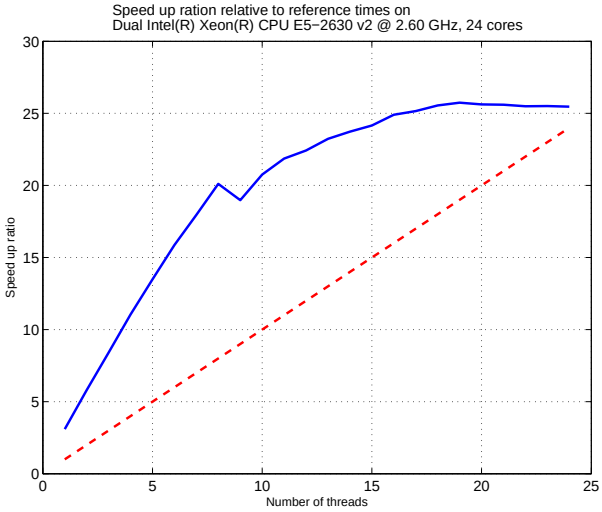

Fig. 2. Attained speed-up for Field IIpro compared to the Field II version 3.22, October 27, 2013 for Dual Intel Xeon E5-2630 2.60 GHz CPUs for a phased array transducer.

The code performance is measured using a point scatterer simulation with the parameters shown in Table I. A 64 element phased array is used to mimic a real imaging situation. Signals from a random collection of one million Gaussian distributed scatterers in front of the transducer is simulated at a range of depths and lateral positions. The scatterers have a depth from 5 to 55 mm and a lateral range from -10 to 10 mm. This ensures that many different spatial impulse responses are simulated for nearly any lateral, elevation, and depth position. Also dynamic focusing and apodization are included in the simulation. The simulation time is measured using the Matlab commands *tic/toc* and the average of 10 simulation runs are shown in the graphs below.

The result of the simulation is shown in Fig. 2. The attained speed-up compared to the public Field II web version 3.22, October 27, 2013 is shown. The x-axis shows the number of threads initiated in the program and the y-axis is the speed-up compared to the web version. The averaged simulation time for 10 realizations for the web free-ware version was 85.1 s. A single thread run for the new version took 27.7 s, a speed-up of 3.1. This is due to the pre-allocation of memory for the innermost part of the spatial impulse response calculation. This is attained when a number of scatterer responses are simulated

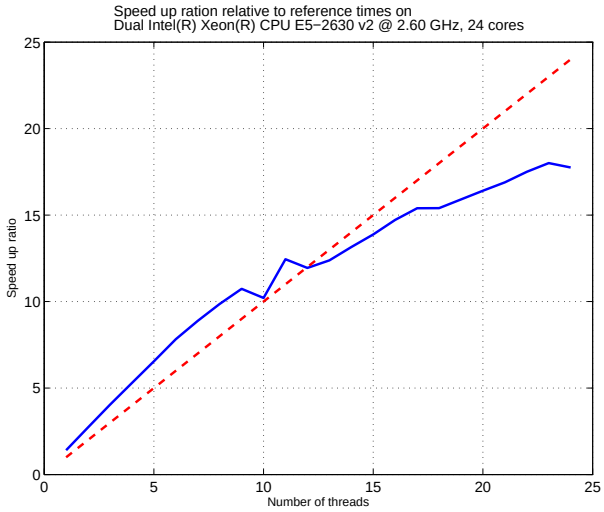

Fig. 3. Attained speed-up for Field IIpro compared to the Field II version 3.22, October 27, 2013 for Dual Intel Xeon E5-2630 2.60 GHz CPUs for a convex array transducer.

as the pre-allocation is only performed once. Employing all 24 cores gave a simulation time of 3.27 s for the one million scatterers corresponding to a speed-up factor of 26 times. For a phantom with 100,000 scatterers, this makes it possible to simulate a full 128 line image in around 42 seconds with full precision. The speed-up is nearly linear until the number of physical cores is reached. After 12 threads the advantage of using more threads tapers off as two hyper-threading cores does not give an increase in speed of a factor of 2.

The red line is Fig. 2 indicates a linear speed-up with the number of threads run. This ideally indicates how using more and more invocations of Matlab would scale a simulation although this probably is optimistic. The new version would always be preferable to using multiple Matlab invocations.

A second example shown in Fig. 3 simulates a 192 element 5 MHz focused convex array transducer with 40 mathematical rectangular elements per physical element. The elevation focus was at 40 mm and the convex radius was 80 mm. One million point scatterers randomly distributed in a plane of 20 x 50 mm (width x depth) with a random Gaussian amplitude were simulated using the command *calc_scat* on the same CPU. The speed-up scales fairly linearly with the number of threads. At 12 threads a speed up of a factor of 13 is attained. After this the speed up tapers of until 24 threads, which attained a factor of 19. This is due to the employment of hyper-threading as the CPU has 12 physical cores and 24 hyper-threaded cores.

The consequence for simulating 1 million point scatterers is shown in Fig. 4, where the simulation time is shown as a function of the number of threads. For one thread it nearly takes 13 minutes, whereas the 24-thread solution cuts the simulation time to 60 seconds. The speed-up in general depends on the transducer, scatterers and simulation, and it varies across applications between 13 and 30.

The parallel execution can also be employed on the solution using bounding lines. The result from this is shown in Fig. 5.

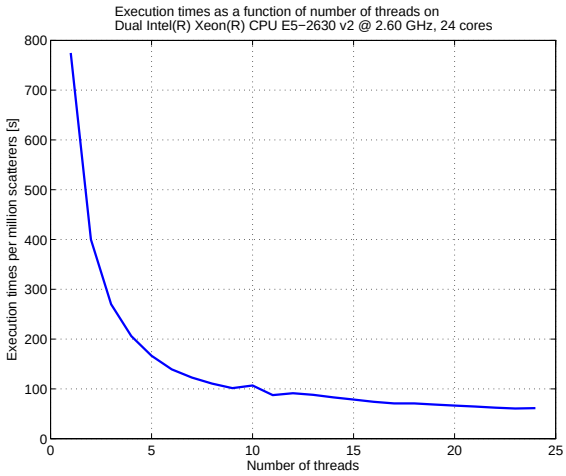

Fig. 4. Time for simulating 1 million point scatterers using a number of threads for Dual Intel Xeon E5-2630 2.60 GHz CPUs.

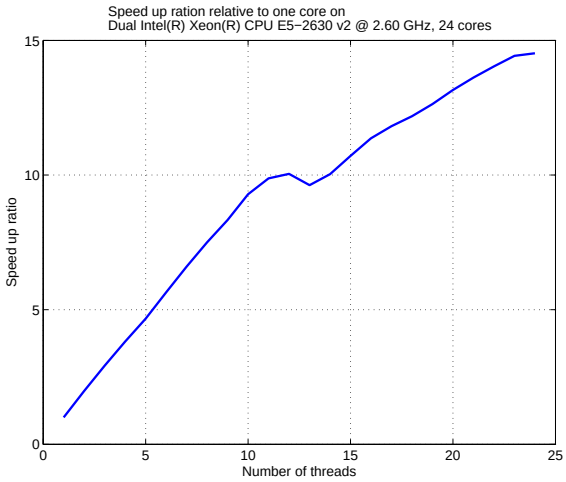

Fig. 5. Speed-up rate compared to running a single thread for the bounding lines method.

The same trend as for the rectangles solution is seen with an increase in speed nearly proportional to the number of threads.

IV. Discussion and Conclusion

Simulation of ultrasound imaging and parameter studies often take a significant number of CPU hours, and they, thus, have to be parallelized to get an efficient development cycle. It is fairly easy to partition the simulation into a number of partial simulations usually as one emission per simulation. Alternatively, the number of scatterers can be partitioned into groups for simulation by a number of computers. The responses then have to be combined after simulation. This involves some work, but the largest drawback is the involvement of a number of Matlab invocations with memory overhead and duplications of transducer and scatterer descriptions in memory. This increases the demand for storage and makes use of the internal CPU cache less efficient. A better solution is therefore to automatically partition the job within the simulation program, although this makes writing the program significantly more difficult. Internal acceleration can be performed using specialized hardware like GPUs, or use the parallel hardware in modern CPUs. The latter is the approach taken in this paper. The performance of the implementation was tested by simulating for a large number of random scatterers for phased and convex array transducers and tabulating the execution time as function of the number of threads employed. A speed-up for the parallel Field IIpro version roughly proportional to the number of real cores was attained. A decrease in performance was seen when the number of threads exceeded the number of physical cores and the CPU had to resort to hyper-threading. The parallel performance gain was attained by using pre-allocation, reuse of memory in the cache, and multi-threaded execution. This also resulted in a speed up of a factor 3.1 compared to the non-optimized version for a single thread execution for the linear array case. The program is fully compatible with older versions, and the new version has been validated and show to yield identical results compared to the old version within machine precision. Only a single command for setting the number of threads to use has been added, and the division of labor is automatically handled by the program. This will accelerate the development cycle without having to write parallel code and having to invoke multiple Matlab sessions.

References

[1] J. P. Asen and S. Holm, "Huygens on speed: Interactive simulation of ultrasound pressure fields," in *Proc. IEEE Ultrason. Symp.*, 2012, pp. 1643–1646.

[2] J. D'hooge, J. Nuyts, B. Bijnens, P. DeMan, P. Suetens, J. Thoen, M. Herregods, and F. VandeWerf, "The calculation of the transient near and far field of a baffled piston using low sampling frequencies," *J. Acoust. Soc. Am.*, vol. 102, no. 1, pp. 78–86, 1997.

[3] L. Tong, A. Ortega, H. Gao, and J. D'hooge, "Fast three-dimensional ultrasound cardiac imaging using multi-transmit beam forming: A simulation study," *Proc. IEEE Ultrason. Symp.*, pp. 1448–1451, 2013.

[4] S. U. Gjerald, R. Brekken, T. Hergum, and J. D'hooge, "Real-time ultrasound simulation using the GPU," in *Proc. IEEE Ultrason. Symp.*, 2012, pp. 258–261.

[5] X. Zeng and R. J. McGough, "Evaluation of the angular spectrum approach for simulations of near-field pressures," *J. Acoust. Soc. Am.*, vol. 123, pp. 68–76, 2007.

[6] G. Pinton and G. Trahey, "Full-wave simulation of finite-amplitude ultrasound in heterogeneous media," in *Proc. IEEE Ultrason. Symp.*, 2007, pp. 130–133.

[7] B. E. Treeby, J. Jaros, A. P. Rendell, and B. T. Cox, "Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method," *J. Acoust. Soc. Am.*, vol. 131, no. 6, pp. 4324–4336, 2012.

[8] J. A. Jensen and N. B. Svendsen, "Calculation of Pressure Fields from Arbitrarily Shaped, Apodized, and Excited Ultrasound Transducers," *IEEE Trans. Ultrason., Ferroelec., Freq. Contr.*, vol. 39, pp. 262–267, 1992.

[9] J. A. Jensen, "Field: A program for simulating ultrasound systems," *Med. Biol. Eng. Comp.*, vol. 10th Nordic-Baltic Conference on Biomedical Imaging, Vol. 4, Supplement 1, Part 1, pp. 351–353, 1996.

[10] G. E. Tupholme, "Generation of acoustic pulses by baffled plane pistons," *Mathematika*, vol. 16, pp. 209–224, 1969.

[11] P. R. Stepanishen, "Transient radiation from pistons in an infinite planar baffle," *J. Acoust. Soc. Am.*, vol. 49, pp. 1629–1638, 1971.

[12] ——, "Pulsed transmit/receive response of ultrasonic piezoelectric transducers," *J. Acoust. Soc. Am.*, vol. 69, pp. 1815–1827, 1981.

[13] J. A. Jensen and P. Munk, "Computer phantoms for simulating ultrasound B-mode and CFM images," in *Acoustical Imaging*, S. Lees and L. A. Ferrari, Eds., vol. 23, 1997, pp. 75–80.

[14] J. A. Jensen and S. Nikolov, "Fast simulation of ultrasound images," in *Proc. IEEE Ultrason. Symp.*, vol. 2, 2000, pp. 1721–1724.